

**DISTRIBUCION PARALELA Y DISTRIBUIDA**

**EDGAR TORRES**

**TEMA**

**ARQUITECTURA CLIENTE SERVIDOR**

**ING.SISTEMAS**

1. **SEMESTRE**

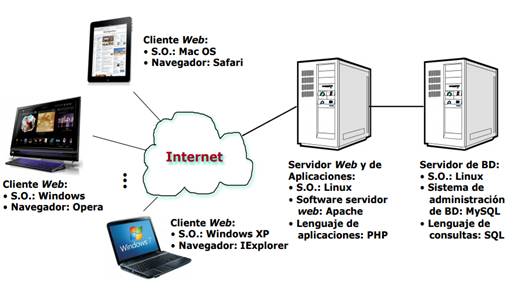
***Arquitectura [cliente](https://www.monografias.com/trabajos11/sercli/sercli.shtml)/servidor***

Según [TIC](https://www.monografias.com/trabajos901/impacto-tic-sociedades-latinoamericanas/impacto-tic-sociedades-latinoamericanas.shtml), (2019)La [estructura](https://www.monografias.com/trabajos15/todorov/todorov.shtml#INTRO) cliente - [servidor](https://www.monografias.com/trabajos12/rete/rete.shtml) es una [arquitectura](https://www.monografias.com/trabajos6/arma/arma.shtml) de [computación](https://www.monografias.com/Computacion/index.shtml) en la que se consigue un procesamiento cooperativo de la [información](https://www.monografias.com/trabajos7/sisinf/sisinf.shtml) por medio de un conjunto de [procesadores](https://www.monografias.com/trabajos5/sisope/sisope.shtml), de tal forma que uno o varios [clientes](https://www.monografias.com/trabajos11/sercli/sercli.shtml), distribuidos geográficamente o no, solicitan [servicios](https://www.monografias.com/trabajos14/verific-servicios/verific-servicios.shtml) de computación a uno o más [servidores](https://www.monografias.com/trabajos12/rete/rete.shtml).

De esta forma, y gracias a esta arquitectura, la totalidad de los procesadores, clientes y servidores, trabajan de forma [cooperativa](https://www.monografias.com/trabajos16/bases-cooperativismo/bases-cooperativismo.shtml#COOPER) para realizar un determinado tratamiento de la información.

Atendiendo a esta visión descentralizada, la arquitectura cliente - servidor consiste en una arquitectura distribuida de computación, en la que las tareas de cómputo se reparten entre distintos procesadores, obteniendo los usuarios finales el resultado final de forma transparente, con [independencia](https://www.monografias.com/trabajos/indephispa/indephispa.shtml) del número de equipos (servidores) que han intervenido en el tratamiento. Se puede decir por tanto que la arquitectura cliente - servidor es un tipo de arquitectura distribuida, posiblemente la más extendida.

**Imagen1: Ejemplo de Arquitectura Cliente/Servidor**



*1Fuente:*[*http*](https://www.monografias.com/trabajos11/wind/wind2.shtml)*://ocw.pucv.cl/cursos-1/arquitectura-de-*[*sistemas*](https://www.monografias.com/trabajos11/teosis/teosis.shtml)*-de-*[*software*](https://www.monografias.com/Computacion/Software/)*/*[*materiales*](https://www.monografias.com/trabajos14/propiedadmateriales/propiedadmateriales.shtml)*-de-clases/*[*web*](https://www.monografias.com/trabajos5/laweb/laweb.shtml)*-cliente-servidor*

***Elementos que forman parte de una arquitectura cliente - servidor***

Un [sistema](https://www.monografias.com/trabajos11/teosis/teosis.shtml) Cliente/Servidor es un [Sistema de Información](https://www.monografias.com/trabajos7/sisinf/sisinf.shtml) distribuido basado en las siguientes características:

* **Servicio:**unidad básica de [diseño](https://www.monografias.com/trabajos13/diseprod/diseprod.shtml). El servidor los proporciona y el cliente los utiliza.
* **Recursos compartidos:**Muchos clientes utilizan los mismos servidores y, a través de ellos, comparten tanto [recursos](https://www.monografias.com/trabajos4/refrec/refrec.shtml) lógicos como físicos.
* **Protocolos asimétricos:**Los clientes inician "conversaciones". Los servidores esperan su establecimiento pasivamente.
* **Transparencia de localización**[**física**](https://www.monografias.com/Fisica/index.shtml)**de los servidores y clientes:**El cliente no tiene por qué saber dónde se encuentra situado el recurso que desea utilizar.
* Independencia de la plataforma HW y SW que se emplee.
* Sistemas débilmente acoplados. [Interacción](https://www.monografias.com/trabajos901/interaccion-comunicacion-exploracion-teorica-conceptual/interaccion-comunicacion-exploracion-teorica-conceptual.shtml) basada en envío de mensajes.
* Encapsulamiento de servicios. Los detalles de la implementación de un [servicio](https://www.monografias.com/trabajos14/verific-servicios/verific-servicios.shtml) son transparentes al cliente.
* Escalabilidad horizontal (añadir clientes) y vertical (ampliar [potencia](https://www.monografias.com/trabajos14/trmnpot/trmnpot.shtml) de los servidores).
* **Integridad:**Datos y [programas](https://www.monografias.com/Computacion/Programacion/) centralizados en servidores facilitan su integridad y [mantenimiento](https://www.monografias.com/trabajos15/mantenimiento-industrial/mantenimiento-industrial.shtml).

En el [modelo](https://www.monografias.com/trabajos/adolmodin/adolmodin.shtml) usual Cliente/Servidor, un servidor, (daemon en la terminología sajona basada en sistemas [UNIX](https://www.monografias.com/trabajos36/sistema-unix/sistema-unix.shtml)/[LINUX](https://www.monografias.com/trabajos14/linux/linux.shtml), traducido como "demonio") se activa y espera las solicitudes de los clientes.

Lo normal es que los servicios de un mismo servidor puedan ser utilizados por múltiples clientes distintos. Tanto los programas cliente como los servidores son con frecuencia parte de un [programa](https://www.monografias.com/Computacion/Programacion/) o aplicación mayores.

**ESQUEMA DE FUNCIONAMIENTO DE UN SISTEMA SEGÚN LA ARQUITECTURA CLIENTE - SERVIDOR.**

El Esquema de funcionamiento de un Sistema Cliente/Servidor sería:

* 1. El cliente solicita una información al servidor.
* 2. El servidor recibe la petición del cliente.
* 3. El servidor procesa dicha solicitud.
* 4. El servidor envía el resultado obtenido al cliente.
* 5. El cliente recibe el resultado y lo procesa.

**COMPONENTES DE LA ARQUITECTURA CLIENTE - SERVIDOR**

El modelo Cliente/Servidor es un modelo basado en la idea del servicio, en el que el cliente es un [proceso](https://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml#PROCE) [consumidor](https://www.monografias.com/trabajos5/comco/comco.shtml#aspe) de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en [función](https://www.monografias.com/trabajos7/mafu/mafu.shtml) del intercambio de mensajes que es el único elemento de acoplamiento entre ambos.

Esta descomposición principalmente consiste en separar los elementos estructurales de esta [tecnología](https://www.monografias.com/Tecnologia/index.shtml) en función de aspectos más funcionales de la misma:

* **Nivel de Presentación:**Agrupa a todos los elementos asociados al componente Cliente.
* **Nivel de Aplicación:**Agrupa a todos los elementos asociados al componente Servidor.
* **Nivel de**[**comunicación**](https://www.monografias.com/trabajos12/fundteo/fundteo.shtml)**:**Agrupa a todos los elementos que hacen posible [la comunicación](https://www.monografias.com/trabajos/lacomunica/lacomunica.shtml) entre los componentes Cliente y servidor.
* **Nivel de**[**base de datos**](https://www.monografias.com/trabajos34/base-de-datos/base-de-datos.shtml)**:**Agrupa a todas las actividades asociadas al acceso de los [datos](https://www.monografias.com/trabajos11/basda/basda.shtml).

***Elementos principales***

**CLIENTE**

Un cliente es todo proceso que reclama servicios de otro. Una definición un poco más elaborada podría ser la siguiente: cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor. Se lo conoce con el término front-end.

Las [funciones](https://www.monografias.com/trabajos7/mafu/mafu.shtml) que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

* Administrar la interfaz de usuario.
* Interactuar con el usuario.
* Procesar la [lógica](https://www.monografias.com/trabajos15/logica-metodologia/logica-metodologia.shtml) de la aplicación y hacer validaciones locales.
* Generar requerimientos de [bases de datos](https://www.monografias.com/trabajos11/basda/basda.shtml).
* Recibir resultados del servidor.
* Formatear resultados.

De este modo el cliente se puede clasificar en:

* **Cliente basado en aplicación de usuario.**Si los datos son de baja interacción y están fuertemente relacionados con la actividad de los usuarios de esos clientes.
* **Cliente basado en lógica de negocio.**Toma datos suministrados por el usuario y/o la base de datos y efectúa los cálculos necesarios según los requerimientos del usuario.

**SERVIDOR**

Un servidor es todo proceso que proporciona un servicio a otros. Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las principales funciones que lleva a cabo el proceso servidor se enumeran a continuación:

* Aceptar los requerimientos de bases de datos que hacen los clientes.
* Procesar requerimientos de bases de datos.
* Formatear datos para trasmitirlos a los clientes.
* Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

**MIDDLEWARE**

El middleware es un módulo intermedio que actúa como conductor entre sistemas permitiendo a cualquier usuario de [sistemas de información](https://www.monografias.com/trabajos7/sisinf/sisinf.shtml)comunicarse con varias [fuentes de información](https://www.monografias.com/trabajos10/formulac/formulac.shtml#FUNC) que se encuentran conectadas por [una red](https://www.monografias.com/Computacion/Redes/). En el caso que nos concierne, es el intermediario entre el cliente y el servidor y se ejecuta en ambas partes.

El middleware se estructura en tres niveles:

* Protocolo de [transporte](https://www.monografias.com/trabajos/transporte/transporte.shtml).
* Network Operating System (NOS).
* Protocolo específico del servicio.

Las principales características de un middleware son:

* Simplifica el proceso de [desarrollo](https://www.monografias.com/trabajos12/desorgan/desorgan.shtml) de aplicaciones al independizar los entornos propietarios.
* Permite la interconectividad de los Sistemas de Información del Organismo.
* Proporciona mayor [control](https://www.monografias.com/trabajos14/control/control.shtml) del negocio al [poder](https://www.monografias.com/trabajos35/el-poder/el-poder.shtml) contar con información procedente de distintas plataformas sobre el mismo soporte.
* Facilita el desarrollo de sistemas complejos con diferentes tecnologías y arquitecturas.

**COMUNICACIÓN ENTRE LOS ELEMENTOS (NOS)**

Como se ha comentado en el apartado anterior, el middleware es un conjunto de aplicaciones encargadas de enlazar al cliente con el servidor. Para ello se estructura en tres capas diferentes:

* Protocolo de transporte: comunes a otras aplicaciones.
* Network Operating System (NOS).
* Protocolo específico del servicio: especiales para distintos tipos de sistemas Cliente/Servidor.

Él NOS es el encargado de proporcionar una apariencia de sistema único a un sistema Cliente/Servidor. Se trata pues, de una extensión del [Sistema Operativo](https://www.monografias.com/Computacion/Sistemas_Operativos/):

* El cliente realiza una llamada a un servicio como si fuera local.
* Él NOS:
* Intercepta la llamada.
* Redirige la llamada al servidor apropiado.
* Devuelve la contestación.

Él NOS debe proporcionar transparencia a los [procesos](https://www.monografias.com/trabajos14/administ-procesos/administ-procesos.shtml#PROCE) Cliente/Servidor con respecto a:

* **Localización:**Los recursos sólo se conocen por su nombre. El sistema en el que se ejecutan es irrelevante.
* **Espacio de nombres:**Las convenciones de los nombres de los recursos deben ser iguales, independientemente del sistema que los soporte.
* **Conexión:**Un único usuario y contraseña para todo el sistema.
* **Replicación:**No se debe diferenciar entre copias de un mismo recurso.
* **Acceso local / remoto:**El acceso a un recurso se debe realizar como si estuviera localizado en el mismo sistema que el programa cliente.
* **Tiempo:**Los relojes de todos los elementos del sistema deben estar sincronizados.
* **Fallos:**El sistema debe proporcionar servicios de detección de fallos, redundancia y reconexión tras un fallo.
* **Administración:**Un único sistema de [gestión](https://www.monografias.com/trabajos15/sistemas-control/sistemas-control.shtml) de todos los recursos.
* **Protocolos:**Idéntica interfaz de [programación](https://www.monografias.com/Computacion/Programacion/) para todos los [protocolos](https://www.monografias.com/trabajos12/mncerem/mncerem.shtml) de transporte.

***Tipos de arquitectura cliente / servidor***

Uno de los aspectos claves para entender la tecnología Cliente/Servidor, y por tanto contar con la capacidad de proponer y llevar a cabo [soluciones](https://www.monografias.com/trabajos14/soluciones/soluciones.shtml) de este tipo, es llegar a conocer la arquitectura de este modelo y los conceptos o ideas asociados al mismo.

Un esquema de clasificación basado en los conceptos de Fat Client/Thin Client, Fat Server/Thin Server, es decir, basado en el tamaño de los componentes. En segundo lugar tenemos una clasificación según la [naturaleza](https://www.monografias.com/trabajos36/naturaleza/naturaleza.shtml) del servicio que nos ofrecen.

**TIPOS DE ARQUITECTURA CLIENTE - SERVIDOR POR TAMAÑO DE COMPONENTES.**

Este tipo de clasificación se basa en los grados de [libertad](https://www.monografias.com/trabajos14/la-libertad/la-libertad.shtml) que brinda el modelo Cliente/Servidor para balancear la carga de proceso entre los niveles de presentación, aplicación y base de datos.

Dependiendo de qué segmento de las capas de software tenga que soportar la mayor o menor carga de procesamiento, se habla de Fat Client (Thin Server) o Fat server (Thin Client).

Consideraciones de este tipo son importantes en el momento de decidir una plataforma de desarrollo, al mismo [tiempo](https://www.monografias.com/trabajos901/evolucion-historica-concepciones-tiempo/evolucion-historica-concepciones-tiempo.shtml) que pueden definir la viabilidad o no de las mismas para enfrentar un cierto número de restricciones impuestas por una problemática a resolver.

**FAT CLIENT (THIN SERVER)**

En este esquema de arquitectura el peso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un [administrador](https://www.monografias.com/trabajos10/habi/habi.shtml) de base de datos.

En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System), y como se concluirá más adelante, tiene pocas posibilidades de aplicarse en sistemas de [misión](https://www.monografias.com/trabajos7/gepla/gepla.shtml) [crítica](https://www.monografias.com/trabajos901/praxis-critica-tesis-doctoral-marx/praxis-critica-tesis-doctoral-marx.shtml).

**FAT SERVER (THIN CLIENT)**

Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el peso de la aplicación corre por el lado del servidor de aplicación.

En general este tipo de arquitectura presenta una flexibilidad mayor para desarrollar una gran variedad de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

**TIPOS DE ARQUITECTURA CLIENTE - SERVIDOR SEGÚN**

**LA NATURALEZA DE SERVICIO PROPORCIONADO.**

**SERVIDORES DE FICHEROS**

Con un servidor de [archivos](https://www.monografias.com/trabajos7/arch/arch.shtml), un cliente lo que hace es requerimientos de los mismos sobre una [red](https://www.monografias.com/Computacion/Redes/). Esta es una forma muy primitiva de servicios de datos, la cual necesita intercambio de muchos mensajes sobre una red para hallar el dato requerido.

Los servidores de archivos usan recursos compartidos sobre la red y son necesarios para crear repositorios de [documentos](https://www.monografias.com/trabajos14/comer/comer.shtml), [imágenes](https://www.monografias.com/trabajos3/color/color.shtml) y archivos grandes sobre la red.

**SERVIDORES DE BASES DE DATOS**

Este [análisis](https://www.monografias.com/trabajos11/metods/metods.shtml#ANALIT) está elaborado desde el punto de vista del modelo Cliente/Servidor, y está directamente relacionado con la arquitectura en dos planos, que se describirá en el apartado siguiente.

Obviamente la creación de aplicaciones Cliente/Servidor está asociada a la utilización de servidores de bases de datos relacionales [SQL](https://www.monografias.com/trabajos14/sqlserver/sqlserver.shtml), y dependiendo de los requerimientos y restricciones se debe elegir entre una arquitectura dos o tres planos.

**SERVIDORES DE TRANSACCIONES**

Estos tipos de sistemas se pueden implementar con cualquiera de las modalidades Cliente/Servidor en dos o tres planos, pero incorporan un elemento principal sobre el cual se elabora y basa toda la fortaleza de este modelo, el [concepto](https://www.monografias.com/trabajos10/teca/teca.shtml) de transacción.

Con un servidor de transacciones el proceso cliente llama a funciones, [procedimientos](https://www.monografias.com/trabajos13/mapro/mapro.shtml) o [métodos](https://www.monografias.com/trabajos11/metods/metods.shtml) que residen en el servidor, ya sea que se trate de un servidor de bases de datos o un servidor de aplicaciones.

**SERVIDORES DE OBJETOS**

Con un servidor de objetos, las aplicaciones Cliente/Servidor son escritas como un conjunto de objetos que se comunican. Los objetos cliente se comunican con los objetos servidores usando un Object Request Broker (ORB). El cliente invoca un [método](https://www.monografias.com/trabajos11/metods/metods.shtml) de un objeto remoto. El ORB localiza el método del objeto en el servidor, y lo ejecuta para devolver el resultado al objeto cliente.

Los servidores de objetos deben soportar concurrencia. La parte central de la comunicación en los servidores de objetos es el ORB:

* Elemento central y principal de esta arquitectura.
* Bus de objetos. Permite la comuniación entre ellos.
* Middleware avanzado: Permite llamadas estáticas y dinámicas a objetos.
* Lenguaje de [descripción](http://monografias.com/trabajos10/anali/anali.shtml) de interfaces independiente del [lenguaje de programación](https://www.monografias.com/Computacion/Programacion/).

**SERVIDORES WEB**

La primera aplicación cliente servidor que cubre todo el planeta es el [World Wide Web](https://www.monografias.com/trabajos32/accesibilidad-world-wide-web/accesibilidad-world-wide-web.shtml). Este nuevo modelo consiste en clientes simples que hablan con servidores Web. Un servidor Web devuelve documentos cuando el cliente pregunta por el nombre de los mismos. Los clientes y los servidores se comunican usando un [protocolo](https://www.monografias.com/trabajos12/mncerem/mncerem.shtml) basado en RPC, llamado HTTP. Este protocolo define un conjunto simple de [comandos](https://www.monografias.com/trabajos7/coman/coman.shtml), los parámetros son pasados como cadenas y no provee [tipos de datos](https://www.monografias.com/trabajos11/basdat/basdat.shtml). La Web y los objetos distribuidos están comenzando a crear un conjunto muy interactivo de computación Cliente/Servidor.

***Modelos cliente/servidor***

Una de las clasificaciones mejor conocidas de las arquitecturas Cliente/Servidor se basa en la idea de planos (tier), la cual es una variación sobre la división o clasificación por tamaño de componentes.

***A nivel de software***

Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura Cliente/Servidor y en la madurez y popularidad de la computación distribuida

**MODELO CLIENTE/SERVIDOR 2 CAPAS**

Esta estructura se caracteriza por la conexión directa entre el proceso cliente y un administrador de bases de datos. Dependiendo de donde se localice el [grupo](https://www.monografias.com/trabajos14/dinamica-grupos/dinamica-grupos.shtml) de tareas correspondientes a la lógica de [negocios](https://www.monografias.com/trabajos15/plan-negocio/plan-negocio.shtml) se pueden tener a su vez dos tipos distintos dentro de esta misma categoría:

**IMPLEMENTADO CON SQL REMOTO**

En este esquema el cliente envía mensajes con solicitudes SQL al servidor de bases de datos y el resultado de cada instrucción SQL es devuelto por la red, no importando si son uno, diez, cien o mil [registros](https://www.monografias.com/trabajos7/regi/regi.shtml). Es el mismo cliente quien debe procesar todos los registros que le fueron devueltos por el servidor de base de datos, según el requerimiento que él mismo hizo.

**Ventajas:**

* Presenta una estructura de desarrollo bastante simple ya que el programador maneja un único [ambiente](https://www.monografias.com/trabajos15/medio-ambiente-venezuela/medio-ambiente-venezuela.shtml) de desarrollo (es más simple respecto al Cliente/Servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

**Inconvenientes:**

* La gran cantidad de información que viaja al cliente congestiona demasiado el tráfico de red, lo que se traduce en bajo rendimiento.
* Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la [construcción](https://www.monografias.com/trabajos35/materiales-construccion/materiales-construccion.shtml) de sistemas no críticos.

**IMPLEMENTADO CON PROCEDIMIENTOS ALMACENADOS**

En este esquema el cliente envía llamadas a funciones que residen en la base de datos, y es ésta quien resuelve y procesa la totalidad de las instrucciones SQL agrupadas en la mencionada función.

**Ventajas:**Presenta las mismas ventajas de una arquitectura dos planos con procedimientos almacenados, pero mejora considerablemente el rendimiento sobre ésta, dado que reduce el tráfico por la red al procesar los datos en la misma base de datos, haciendo viajar sólo el resultado final de un conjunto de instrucciones SQL.

**Inconvenientes:**Si bien la complejidad de desarrollo se ve disminuida, se pierde flexibilidad y escalabilidad en las soluciones implantadas. Obliga a basar el peso de la aplicación en SQL extendido, propios del proveedor de la base de datos que se elija. Debiera considerarse que sí bien los procedimientos almacenados (stored procedures), los desencadenantes (triggers) y las reglas (constraint) son útiles, en rigor son ajenos al estándar de SQL

**MODELO CLIENTE/SERVIDOR 3 CAPAS**

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la carga de [trabajo](https://www.monografias.com/trabajos34/el-trabajo/el-trabajo.shtml) entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

**Ventajas:**

* Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto a la estructura en dos planos).
* Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones. Provee escalabilidad horizontal y vertical.
* Se mantiene la independencia entre el [código](https://www.monografias.com/trabajos12/eticaplic/eticaplic.shtml) de la aplicación (reglas y [conocimiento](https://www.monografias.com/trabajos/epistemologia2/epistemologia2.shtml) del negocio) y los datos, mejorando la portabilidad de las aplicaciones.

**Inconvenientes:**

* Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con Cliente/Servidor dos planos.
* Existen pocos [proveedores](https://www.monografias.com/trabajos6/lacali/lacali.shtml#influencia) de [herramientas](https://www.monografias.com/trabajos11/contrest/contrest.shtml) integradas de desarrollo con relación al modelo Cliente/Servidor dos planos, y normalmente son de alto [costo](https://www.monografias.com/trabajos7/coad/coad.shtml#costo).

***A nivel de hardware***

Esta clasificación del modelo Cliente/Servidor se basa igualmente en la [distribución](https://www.monografias.com/trabajos11/travent/travent.shtml) de los procesos y elementos entre sus componentes, pero centrándose en la parte física del mismo, en el que [la administración](https://www.monografias.com/Administracion_y_Finanzas/index.shtml) de la interfaz gráfica se asocia a los clientes PC y la [seguridad](https://www.monografias.com/trabajos/seguinfo/seguinfo.shtml) e integridad de los datos quedan asociados a ambientes mainframe o por lo menos a servidores locales y/o centrales.

**MODELO CLIENTE / SERVIDOR 2 CAPAS**

Los clientes son conectados vía [LAN](https://www.monografias.com/trabajos15/redes-clasif/redes-clasif.shtml) a un servidor de aplicaciones local, el cual, dependiendo de la aplicación puede dar acceso a los datos administrados por él.

**MODELO CLIENTE / SERVIDOR 3 CAPAS**

Los clientes son conectados vía LAN a un servidor de aplicaciones local, el cual a su vez se comunica con un servidor central de bases de datos. El servidor local tiene un [comportamiento](https://www.monografias.com/trabajos16/comportamiento-humano/comportamiento-humano.shtml) dual, dado que actúa como cliente o servidor en función de la [dirección](https://www.monografias.com/trabajos15/direccion/direccion.shtml) de la comunicación.

***¿Qué significa [TCP/IP?](https://www.monografias.com/trabajos5/datint/datint.shtml" \l "pro)***

Según Vialfa, (2017) TCP/IP es un conjunto de protocolos que permiten la comunicación entre los ordenadores pertenecientes a una red. La sigla TCP/[IP](https://www.monografias.com/trabajos/protocolotcpip/protocolotcpip.shtml) significa Protocolo de control de transmisión/Protocolo de Internet y se pronuncia "T-C-P-I-P". Proviene de los nombres de dos protocolos importantes incluidos en el conjunto TCP/IP, es decir, del protocolo TCP y del protocolo IP.

En algunos aspectos, TCP/IP representa todas las reglas de comunicación para [Internet](https://www.monografias.com/Computacion/Internet/) y se basa en la noción de dirección IP, es decir, en la idea de brindar una dirección IP a cada equipo de la red para poder enrutar paquetes de datos. Debido a que el conjunto de protocolos TCP/IP originalmente se creó con fines militares, está diseñado para cumplir con una cierta cantidad de criterios, entre ellos, dividir mensajes en paquetes, usar un sistema de direcciones, enrutar datos por la red y detectar errores en las transmisiones de datos.

El conocimiento del conjunto de protocolos TCP/IP no es esencial para un simple usuario, de la misma manera que un espectador no necesita saber cómo funciona su red audiovisual o de [televisión](https://www.monografias.com/trabajos37/historia-television/historia-television.shtml). Sin embargo, para las personas que desean administrar o brindar soporte técnico a una red TCP/IP, su conocimiento es fundamental.

***La diferencia entre estándar e implementación***

En general, TCP/IP relaciona las siguientes dos nociones:

La noción de **estándar**: el protocolo TCP/IP representa la manera en la que se realizan las [comunicaciones](https://www.monografias.com/trabajos/lacomunica/lacomunica.shtml) en una red.

La noción de **implementación**: la designación TCP/IP generalmente se extiende a software basado en el protocolo TCP/IP. En realidad, TCP/IP es un modelo cuya aplicación de red utilizan los desarrolladores. Las aplicaciones son, por lo tanto, implementaciones del protocolo TCP/IP.

TCP/IP ES UN MODELO DE CAPAS

Para poder aplicar el modelo TCP/IP en cualquier equipo, es decir, independientemente del sistema operativo, el sistema de protocolos TCP/IP se ha dividido en diversos módulos. Cada uno de éstos realiza una tarea específica. Además, estos módulos realizan sus tareas uno después del otro en un orden específico, es decir que existe un sistema estratificado. Ésta es la razón por la cual se habla de **modelo de capas**.

LA IMPORTANCIA DE UN SISTEMA DE CAPAS

El [objetivo](https://www.monografias.com/trabajos16/objetivos-educacion/objetivos-educacion.shtml) de un sistema en capas es dividir el problema en diferentes partes (las capas), de acuerdo con su nivel de abstracción.

Cada capa del modelo se comunica con un nivel adyacente (superior o inferior). Por lo tanto, cada capa utiliza los servicios de las capas inferiores y se los proporciona a la capa superior.

EL MODELO OSI

El [modelo OSI](https://www.monografias.com/trabajos13/modosi/modosi.shtml) es un modelo que comprende 7 capas, mientras que el modelo TCP/IP tiene sólo 4. En realidad, el modelo TCP/IP se desarrolló casi a la par que el modelo [OSI](https://www.monografias.com/trabajos13/modosi/modosi.shtml). Es por ello que está influenciado por éste, pero no sigue todas las especificaciones del modelo OSI. Las capas del modelo OSI son las siguientes:

* **La capa física**define la manera en la que los datos se convierten físicamente en [señales](https://www.monografias.com/trabajos36/signos-simbolos/signos-simbolos.shtml) digitales en [los medios de comunicación](https://www.monografias.com/trabajos14/medios-comunicacion/medios-comunicacion.shtml) (pulsos eléctricos, [modulación](https://www.monografias.com/trabajos10/modul/modul.shtml) de [luz](https://www.monografias.com/trabajos5/natlu/natlu.shtml), etc.).
* **La capa de enlace de datos**define la interfaz con la tarjeta de interfaz de red y cómo se comparte el medio de transmisión.
* **La capa de red**permite administrar las direcciones y el enrutamiento de datos, es decir, su ruta a través de la red.
* **La capa de transporte**se encarga del transporte de datos, su división en paquetes y la [administración](https://www.monografias.com/Administracion_y_Finanzas/index.shtml) de potenciales errores de transmisión.
* **La capa de sesión**define el inicio y la finalización de las sesiones de comunicación entre los equipos de la red.
* **La capa de presentación**define el formato de los datos que maneja la capa de aplicación (su representación y, potencialmente, su compresión y cifrado) independientemente del sistema.
* **La capa de aplicación**le brinda aplicaciones a la interfaz. Por lo tanto, es el nivel más cercano a los usuarios, administrado directamente por el software.

***El modelo TCP/IP***

El modelo TCP/IP, influenciado por el modelo OSI, también utiliza el enfoque modular (utiliza módulos o capas), pero sólo contiene cuatro: acceso a la red, Internet, transporte y aplicación.

Como puede verse, las capas del modelo TCP/IP tienen tareas mucho más diversas que las del modelo OSI, considerando que ciertas capas del modelo TCP/IP se corresponden con varios niveles del modelo OSI.

Las funciones de las diferentes capas son las siguientes:

**Capa de acceso a la red**: específica la forma en la que los datos deben enrutarse, sea cual sea el tipo de red utilizado.

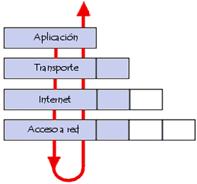
**Capa de Internet**: es responsable de proporcionar el paquete de datos (datagrama).

**Capa de transporte**: brinda los datos de enrutamiento, junto con los mecanismos que permiten conocer [el estado](https://www.monografias.com/trabajos12/elorigest/elorigest.shtml) de la transmisión. Comprende a los protocolos TCP y UDP.

**Capa de aplicación**: incorpora aplicaciones de red estándar ([Telnet](https://www.monografias.com/trabajos7/tcp/tcp.shtml#Telnet), SMTP, [FTP](https://www.monografias.com/trabajos/ftpbasico/ftpbasico.shtml), etc.).

ENCAPSULACIÓN DE DATOS

Durante una transmisión, los datos cruzan cada una de las capas en el nivel del equipo remitente. En cada capa, se le agrega información al paquete de datos. Esto se llama **encabezado**, es decir, una recopilación de información que garantiza la transmisión. En el nivel del equipo receptor, cuando se atraviesa cada capa, el encabezado se lee y después se elimina. Entonces, cuando se recibe, el mensaje se encuentra en su [estado](https://www.monografias.com/trabajos12/elorigest/elorigest.shtml) original:



**Imagen1: Ejemplo de Arquitectura Cliente/Servidor**

*2Fuente: http://es.ccm.net/contents/282-tcp-ip*

En cada nivel, el paquete de datos cambia su aspecto porque se le agrega un encabezado. Por lo tanto, las designaciones cambian según las capas: el paquete de datos se denomina **mensaje**en el nivel de la capa de aplicación. El mensaje después se encapsula en forma de **segmento**en la capa de transporte. Una vez que se encapsula el segmento en la capa de Internet, toma el nombre de **datagrama**. Finalmente, se habla de **trama**en el nivel de capa de acceso a la red.

***Conclución***

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

Esta arquitectura se basa en la existencia de dos tipos de aplicaciones ejecutándose de forma independiente

 Una de las aplicaciones actúa como servidora la otra como cliente.

 El cliente pide datos, se envían en forma de consulta al servidor el servidor procesa la consulta y devuelve los datos al cliente y solo viajan los datos pedidos.

Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, [redes](https://www.monografias.com/Computacion/Redes/), hardware